(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 686 906 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **95303720.7**

(51) Int. Cl.[6]: **G06F 1/00**

(22) Date of filing: **31.05.95**

(30) Priority: **10.06.94 US 258244**

(43) Date of publication of application:
13.12.95 Bulletin 95/50

(84) Designated Contracting States:
DE FR GB NL SE

(71) Applicant: **SUN MICROSYSTEMS, INC.**
2550 Garcia Avenue
Mountain View, CA 94043 (US)

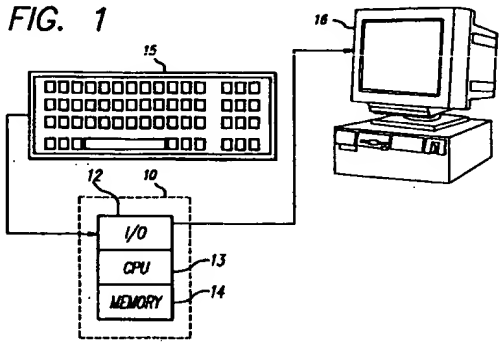
(72) Inventor: **Chang, Sheue-Ling**
22345 Regnart Road
Cupertino, California 95014 (US)
Inventor: **Gosling, James**
P.O. Box 620509
Woodside,
California 95014 (US)

(74) Representative: **Wombwell, Francis et al**
Potts, Kerr & Co.
15, Hamilton Square
Birkenhead
Merseyside L41 6BR (GB)

(54) Method and apparatus for enhancing software security and distributing software

(57) Source code to be protected, a software application writer's private key, along with an application writer's license provided to the first computer. The application writer's license includes identifying information such as the application writer's name as well as the application writer's public key. A compiler program executed by the first computer compiles the source code into binary code, and computes a message digest for the binary code. The first computer then encrypts the message digest using the application writer's private key, such that the encrypted message digest is defined as a digital "signature" of the application writer. A software passport is then generated which includes the application writer's digital signature, the application writer's license and the binary code. The software passport is then distributed to a user using any number of software distribution models known in the industry. A user, upon receipt of the software passport, loads the passport into a computer which determines whether the software passport includes the application writer's license and digital signature. In the event that the software passport does not include the application writer's license, or the application writer's digital signature, then the user's computer system discards the software passport and does not execute the binary code. As an additional security step, the user's computer computes a second message digest for the software passport and compares it to the first message digest, such that if the first and second message digests are not equal, the software passport is also rejected by the user's computer and the code is not executed. If the first and second message digests are equal, the user's computer extracts the application writer's public key from the application writer's license for verification. The application writer's digital signature is decrypted using the application writer's public key. The user's computer then compares a message digest of the binary code to be executed, with the decrypted application writer's digital signature, such that if they are equal, the user's computer executes the binary code.

FIG. 1



EP 0 686 906 A2

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates to the use of public key encryption, and more particularly, the present invention relates to the use of public key encryption to achieve enhanced security and product authentication in the distribution of software.

2. Art Background:

Public key encryption is based on encryption algorithms that have two keys. One key used for encryption, and the other key is used for decryption. There is a known algorithm that computes the second key given the first. However, without full knowledge of all the parameters, one cannot compute the first key given the second key. The first key is referred to as the "private key", and the second key is referred to as the "public key". In practice, either the private key or the public key may be used to encrypt a message, with the opposite key used to decrypt it. In general, the private key must be kept private, but the public key may be provided to anyone. A variety of public key cryptographic schemes have been developed for the protection of messages and data (See, Whitfield Diffie, "The First Ten Years of Public Key Cryptography" (IEEE Proceedings, Vol. 76, No. 5, 1988) and Fahn, "Answers to Frequently Asked Questions about Today's Cryptography (RSA Laboratories 1992).

Public key cryptography is used to send secure messages across public communication links on which an intruder may eavesdrop, and solves the problem of sending the encryption password to the other side securely.

Public key systems may also be used to encrypt messages, and also to effectively sign messages, allowing the received party to authenticate the sender of the message. One can also use public key cryptography to seal or render tamperproof a piece of data. In such event, the sender computes a message digest from the data using specially designed cryptographically strong digests designed for this purpose. The sender then uses the private key to encrypt the message digest, wherein this encrypted message digest is called a digital "signature". The sender then packages the data, the message digest and the public key together. The receiver may check for tampering by computing the message digest again, then decrypting the received message digest with the public key. If the recomputed and decrypted message digests are identical, there was no tampering of the data.

"Viruses" and "worms" are computer code cleverly inserted into legitimate programs which are subsequently executed on computers. Each time the program is executed the virus or worm can cause damage to the system by destroying valuable information, and/or further infect and spread to other machines on the network. While there are subtle differences between a virus and a worm, a critical component for both is that they typically require help from an unsuspecting computer user to successfully infect a computer or a corporate network.

Infection of computers by viruses and worms is a general problem in the computer industry today. In addition, corporate networks are vulnerable to frontal assaults, where an intruder breaks into the network and steals or destroys information. Security breaches of any kind on large corporate networks are a particularly worrisome problem, because of the potential for large-scale damage and economic loss. Moreover, security breaches are more easily accomplished when a corporate network is connected to a public network, such as the Internet. Companies take a variety of measures to guard against breaches of network security, either through frontal assaults or infections, without cutting themselves off from the benefits of being connected to a world-wide network.

The solution adopted by most companies that wish to reap the benefits of connecting to the Internet, while maintaining security, is the installation of a firewall. Firewalls generally restrict Internet file transfers and telnet connections. Such transfers and connections can only be initiated from within the corporate network, such that externally initiated file transfers and telnet connections are refused by the firewall. Firewalls allow electronic mail and network news to freely flow inside the firewall's private network. The use of corporate firewalls allows employees to readily exchange information within the corporate environment, without having to adopt extreme security measures. A good firewall implementation can defend against most of the typical frontal assaults on system security.

One method of preventing viruses and worms from infecting a corporate network is to never execute a program that may contain viruses. In general, programs legitimately deployed throughout the corporate network should be considered virus free. All binary executables, all unreviewed shell scripts, and all source code fetched from outside the firewall are software that may contain a worm or virus.

However, outside binary executables, shell scripts, and source code may enter a corporate firewall through an E-mail attachment. For example, the shell scripts that are used to make and send multiple files using E-mail and the surveytools that

start up by activating the E-mail attachment may allow virus entry. Executables can also be directly fetched through the iftp program, through a world-wide web browser such as Mosaic, or from an outside contractor whose network has already been compromised.

In addition, the commercial software release and distribution process presents security and authentication problems. For example, some of the information associated with software, such as the originating company or author, restricted rights legends, and the like are not attached to the code itself. Instead, such information is provided as printed matter, and is separated from the code once the package is opened and the code installed. Even applications that attempt to identify themselves on start-up are susceptible to having the identification forged or otherwise counterfeited.

A user has no mechanism to authenticate that the software sold is actually from the manufacturer shown on the label. Unauthorized copying and the sale of software is a significant problem, and users who believe that they are buying software with a manufacturer's warranty instead purchase pirated software, with neither a warranty nor software support. The problem of authenticating the original source of the software is accentuated when software is intended to be distributed through networks, and a user's source for the software may be far removed from the original writer of the software. In addition, a user does not have that ability to verify that the software purchased contains only the original manufacturer's code. A user also does not have a method for detecting any tampering, such as the existence of a virus, that may cause undesirable effects.

All of the above problems are related to the transport of software both from manufacturers to users and from user to user. Furthermore, the transport problem is independent of the transport medium. The problem applies to all transport media, including floppy disk, magnetic tape, CD-ROM and networks.

As will be described, the present invention provides a method and apparatus for authenticating that software distributed by a manufacturer is a legitimate copy of an authorized software release, and that the software contains only the original manufacturers code without tampering. The present invention solves the above identified problems through the use of a "software passport" which includes the digital signature of the application writer and manufacturer. As will be described, the present invention may also be used to protect intellectual property, in the form of copyrighted computer code, by utilizing cryptographic techniques referred to herein as public key encryption.

SUMMARY OF THE INVENTION

This invention provides a method and apparatus utilizing public key encryption techniques for enhancing software security and for distributing software. The present invention includes a first computer which is provided with source code to be protected using the teachings of the present invention. In addition, a software application writer's private key, along with an application writer's license provided to the first computer. An application writer generally means a software company such as Microsoft Corporation, Adobe or Apple Computer, Inc. The application writer's license includes identifying information such as the application writer's name as well as the application writer's public key. A compiler program executed by the first computer compiles the source code into binary code, and computes a message digest for the binary code. The first computer then encrypts the message digest using the application writer's private key, such that the encrypted message digest is defined as a digital "signature" of the application writer. A software passport is then generated which includes the application writer's digital signature, the application writer's license and the binary code. The software passport is then distributed to a user using any number of software distribution models known in the industry.

A user, upon receipt of the software passport, loads the passport into a computer which determines whether the software passport includes the application writer's license and digital signature. In the event that the software passport does not include the application writer's license, or the application writer's digital signature, then the user's computer system discards the software passport and does not execute the binary code. As an additional security step, the user's computer computes a second message digest for the software passport and compares it to the first message digest, such that if the first and second message digests are not equal, the software passport is also rejected by the user's computer and the code is not executed. If the first and second message digests are equal, the user's computer extracts the application writer's public key from the application writer's license for verification. The application writer's digital signature is decrypted using the application writer's public key. The user's computer then compares a message digest of the binary code to be executed, with the decrypted application writer's digital signature, such that if they are equal, the user's computer executes the binary code. Accordingly, software products distributed with the present invention's software passport permits the user's computer to authenticate the software as created by an authorized application writer

who has been issued a valid application writer's license. Any unauthorized changes to the binary code comprising the distributed software is evident through the comparison of the calculated and encrypted message digests.

The present invention is also described with reference to an embodiment used by computing platforms designed to execute only authorized software. A platform builder provides an application writer with a platform builder's digital signature which is included in the application writer's license. The first computer compiles the software into binary code and computes a first message digest for the binary code. The first computer further encrypts the first message digest using the application writer's private key, such that the encrypted first message digest is defined as the application writer's digital signature. A software passport is generated which includes the application writer's digital signature, the application writer's license and the binary code. The software passport is then distributed to a user through existing software distribution channels. The user's computing platform, which may be a computer, a video game box or a set top box, is provided with the platform builder's public key. Upon receipt of the software passport, the computing platform determines if the software passport includes an application writer's license. If it does not, the hardware platform rejects the execution of the code. If a software passport is present, the hardware platform extracts the application writer's license from the passport and determines whether or not the passport includes the platform builder's signature. The platform builder's signature is then decrypted using the public key provided in the platform. The computing platform recomputes the message digest of the application writer's license, and compares the received message digest with the recomputed message digest, such that if the digests are not equal, the software passport is not considered genuine and is rejected. If the message digests are equal, the hardware platform extracts the application writer's public key from the application writer's license, and extracts the application writer's digital signature. The hardware platform then recomputes the message digest of the binary code comprising the application software to be executed, and decrypts the application writer's digital signature using the application writer's public key. The hardware platform then compares the recomputed message digest for the binary code with the application writer's decrypted signature, such that if they are equal, the binary code is executed by the hardware platform. If the recomputed message digest and the application writer's decrypted signature are not equal, the software passport is rejected and the code is not executed.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a data processing system incorporating the teachings of the present invention.

Figure 2 conceptually illustrates use of the present invention's software passport where the application code and the software passport are provided in separate files.

Figure 3 conceptually illustrates use of the present invention's use of the software passport where the application code and the software passport are distributed in the same file.

Figure 4 diagrammatically illustrates the present invention's process for generating a software passport.

Figure 5 diagrammatically illustrates the use of the present invention for platform producer licensing.

Figures 6a and 6b are flowcharts illustrating the steps executed by the present invention for verifying that a valid software license exists, and that the software writer's ("SW's") signature is valid, prior to permitting the execution of a computer program.

Notation and Nomenclature

The detailed descriptions which follow are presented largely in terms of symbolic representations of operations of data processing devices. These process descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities may take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, displayed and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, messages, names, elements, symbols, operations, messages, terms, numbers, or the like. It should be borne in mind, however, that all of these similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

In the present invention, the operations referred to are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers or other similar devices. In all cases, the reader is advised to keep in mind the distinction between the method operations of operating a computer and the method

of computation itself. The present invention relates to method steps for operating a computer, coupled to a series of networks, and processing electrical or other physical signals to generate other desired physical signals.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purposes or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. The method/process steps presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein, or it may prove more convenient to construct specialized apparatus to perform the required method steps. The required structure for a variety of these machines will be apparent from the description given below.

Detailed Description of the Invention

In the following description, numerous specific details are set forth such as system configurations, representative data, computer code organization, encryption methods, and devices, etc., to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well known circuits and structures are not described in detail in order to not obscure the present invention. Moreover, certain terms such as "knows", "verifies", "compares", "examines", "utilizes", "finds", "determines", "challenges", "authenticates", etc., are used in this Specification and are considered to be terms of art. The use of these terms, which to a casual reader may be considered personifications of computer or electronic systems, refers to the functions of the system as having human-like attributes, for simplicity. For example, a reference herein to an electronic system as "determining" something is simply a shorthand method of describing that the electronic system has been programmed or otherwise modified in accordance with the teachings herein. The reader is cautioned not to confuse the functions described with everyday human attributes. These functions are machine functions in every sense.

Exemplary Hardware

Figure 1 illustrates a data processing system in accordance with the teachings of the present invention. Shown is a computer 10, which comprises three major components. The first of these is an input/output (I/O) circuit 12 which is used to communicate information in appropriately structured form to and from other portions of the computer 10. In addition, computer 10 includes a central processing (CPU) 13 coupled to the I/O circuit 12 and a memory 14. These elements are those typically found in most general purpose computers and, in fact, computer 10 is intended to be representative of a broad category of data processing devices. Also, the computer 10 may be coupled to a network, in accordance with the teachings herein. The computer 10 may further include encrypting and decrypting circuitry incorporating the present invention, or as will be appreciated, the present invention may be implemented in software executed by computer 10. A raster display monitor 16 is shown coupled to the I/O circuit 12 and issued to display images generated by CPU 13 in accordance with the present invention. Any well known variety of cathode ray tube (CRT) or other type of display may be utilized as display 16.

The present invention's software passport identifies a portion of software, or some machine code (hereinafter "code"), in a manner similar to how a physical passport identifies a person. The concept is similar to the real-life passport system which forms the basis of a trust model among different nations. Physical passports enable border entry officers to identify each individual and make certain decisions based on his/her passport. As will be described below, a software passport is a modern release process for distributing software products. A software passport gives a software product an identity and a brand name. The software passport provides the basis of a trust model and allows computer users to identify and determine the genuineness of a software product based on the information contained in its passport.

Referring now to **Figure 2**, the present invention is illustrated in conceptual form for the case where the computer code (comprising a piece of software) and the software passport are in separate files. **Figure 3** illustrates the use of the present invention where the computer code comprising a piece of software and the software passport are in the same file.

As illustrated in **Figures 2** and **3**, the information included in the present invention's software passport may include:

product information, such as the software product's name and any other relevant information to the specific product;

company information including the name of the company or the software application writer who has produced the product;

a validity date which includes the issue date of the software passport and the expiration date of the passport;

a restricted rights legend including copyright

notices and other similar legends;

the software code body including executable application code distributed to the user;

an application writer's license; and,

a software application writer's digital signature.

It will be appreciated that the components of a software passport are generally self-explanatory, with the application writer's license and digital signature explained in more detail below.

SOFTWARE PRODUCER'S DIGITAL SIGNATURE

A digital "signature" is produced by using certain cryptographic techniques of computing a message digest of a piece of software code (hereinafter "code"), and encrypting the message digest using the signer's private key. There are many known message digest algorithms, such as the MD2, MD4, and MD5 algorithms published by RSA, Inc. The use of private cryptographic techniques makes this signature very difficult to forge since the signer keeps the private key secret. The reader is referred to the papers by Whitfield Diffie, "The First Ten Years of Public Key Cryptography", Vol. 76, No. 5 (IEEE Proceedings, May 1988); and Whitfield Diffie, et al., "Authentication and Authenticated Key Exchanges" (1992 Kluwer Academic Publishers) incorporated herein by reference, for a detailed description of the operation of Diffie-Helman certificates and public key cryptography.

One may conceptualize the computing of the message digest for a piece of code as a mechanism of taking a photo snapshot of the software. When the code changes, its message digest reflects any differences. In the system of the present invention, this "digital signature" is stamped on the product prior to its release. The digital signature associates a product with the entity that has produced it, and enables consumers to evaluate the quality of a product based on the reputation of the producer. The signature also permits a consumer to distinguish the genuineness of a product.

SOFTWARE PRODUCER'S LICENSE

The present invention's software producer's license (at time referred to herein as the "application writer's license") is an identification similar to the home repair `contractor's license issued by a state. A software producer's license identifies and certifies that the producer is authorized to perform certain software production activities. It is contemplated that the software producer's license will be issued by some commonly-trusted authority established by the computer software industry. Before issuing an license to a software producer, this authority performs a defined process to authenticate the person or company, and to verify their job skill; as a state does before issuing a contractor's license. For convenience, in this Specification, this commonly-trusted entity is referred to as the Software Publishing Authority ("SPA").

A software producer's license contains the following information:

the producer's name;

the license's issue date;

the license's expiration date;

the producer's public key;

the name of the issuing authority, SPA; and

the SPA's digital signature.

A software producer's license associates an application writer with a name and a public key. It enables a software producer to produce multiple products, and to sign every product produced. The public key embedded in a license belongs to the person who owns the license. This public key can later be used by any third party to verify the producer's digital signature. A user who has purchased a product can determine the genuineness of a product by using the public key embedded in the producer's identification to authenticate the digital signature.

The SPA's digital signature is generated by computing the message digest of the producer's identification and encrypting the message digest using the SPA's private key. Since the SPA's private key is kept private to the SPA, third parties are not able to easily forge the SPA's signature to produce a fake identification.

In accordance with the teachings of the present invention, a software application writer ("SW") supplies three major pieces of information to a compiler prior to compilation of the code:

the source code written by the application writer;

the application writer's private key; and

the application writer's license.

The code included in a passport may comprise source code in various computer languages, assembly code, machine binary code, or data. The code may be stored in various formats. For example, a piece of source code may be stored in a clear text form in the passport. A portion of binary executable machine code may also be stored in a compacted format in the passport, using certain well known compaction algorithms such as Huffman encoding. The format used in a particular implementation is indicated by a flag in the passport.

Binary executable code may further be stored in a printable-character set format to allow the passport to be printed. A user would then reverse the printable-format to recover the software. Moreover, code protected by intellectual property, such as copyright or patent, may be stored in an encrypted format in the passport. In such case, it is

contemplated that a user may be required to pay a license fee prior to gaining access to the software.

Referring now to **Figure 4**, to generate the software passport of the present invention, the original source code 20, the application writer's private key 22, and the application writer's license 24 is provided to a compiler 26. As illustrated, the application writer's license 24 includes the writer's name 30, the writer's public key 32 and a validity date 34.

The compiler 26 then compiles the source code 20 into binary code. The compiler 26 further computes the message digest of the binary code, and encrypts the message digest using the private key 22 supplied by the application writer. This encrypted message digest constitutes the application writer's signature.

A digital signature of the application writer is produced and embedded in the passport. The compiler 26 also embeds the application writer's license 24 in the passport. The application writer's license 24 allows any user who has purchased the product to recognize the maker of the product. The application writer's digital signature in the passport allows any user to verify the genuineness of the product. The SPA's digital signature in the application writer's license 24 provides the user with the ability to verify that an application writer is a licensed application writer by using SPA's public key to encrypt the signature.

As shown in **Figure 4**, the generated software passport 38, including the application code is then distributed using any desired software distribution model. The passport 38 is received by a user and is executed using an operating system (OS) running on a computer system ("platform") such as the system of **Figure 1**.

Referring now to **Figure 5**, the use of the present invention by platform builders will be described. In the electronic game industry and the interactive television cable set-top box industry, platform producers often desire to allow only authorized code to be executed on their particular platform. To be able to control the accessibility of a platform, the received code must be identifiable and the platform must be able to identify the software when it arrives. As illustrated in **Figure 5**, the present invention may be applied in a platform producer licensing scheme with particular application for use in set-top box and video game environments.

Referring now to **Figures 6a** and **6b**, a platform producer may issue a "programmer's license" to a set of application writers (alternatively referred to as "software writers") who are authorized to write application code for a particular platform. A programmer's license issued by a platform producer is similar to the programmer's identification issued by the SPA, except that the license is digitally signed by the platform producer instead of by the SPA. The programmer's license contains the following information:

the producer's name;
the issue license data;
the license expiration date;
the producer's public key;
the issuing authority (the platform producer); and
the platform producer's digital signature.

The platform producer's digital signature is generated by computing the message digest of the license, and encrypting the message digest using the platform producer's private key.

The software produced by a licensed application writer will include a valid passport 50 (see **Figures 5** and **6a**) which contains a genuine writer's digital signature, and a valid application writer's license 52 issued by the platform builder. Any application writer who is not authorized by the platform builder will not possess a valid license. Therefore, the software passport generated by an unauthorized person will either have no valid license or no valid signature.

The public key 54 of the platform builder is embedded in the platform (e.g., video game) for the verification process. At execution time, the platform extracts the public key 54 embedded in the system to verify that a passport contains a valid application writer's license 52. The digital signature in the application writer's license is generated by computing the message digest of the license 52 and encrypting the message digest using the platform builder's private key. The system of the present invention can thus recover the original message digest by decrypting the signature using the platform builder's public key 54. The verification process of the application writer's license may be accomplished by:

1. recomputing the message digest of the application license 52 in the passport 50,
2. recovering the original message digest, and
3. comparing the old digest with the newly computed digest.

The passport 50 contains a valid application writer's license if the two message digests are the same. Otherwise the license is not valid. The verification process of the present invention is illustrated in the flow chart of **Figure 6(a)**.

It will be appreciated that even if the passport 50 does contain a valid application writer's license, the application writer might have stolen the license by copying it from some other authorized writer's passport. In this case, the unauthorized writer would not have a correct private key 58 to forge the signature of the authorized writer. It is contemplated that the system will further verify the

signature of the application writer 60. It will be recalled that the application writer's digital signature in the passport was generated by computing the message digest of the passport and encrypting the message digest using the application writer's private key 58. The original message digest may be recovered by decrypting the signature using the writer's public key 62 embedded in the application writer's license 52, which is embedded in the passport 50. The application writer's digital signature may then be verified by:

1. recomputing the message digest of the passport 50,
2. recovering the original message digest, and
3. comparing the old digest with the new digest.

The signature is valid if the two message digests are the same. Otherwise the passport is not valid and the platform will reject the execution of the software. The steps executed by the present invention to verify the application writer's digital signature are illustrated in flow chart for **Figure 6-(b)**.

It will be further noted that the security scheme of the present invention may be used to protect inventions and authorship protected by intellectual property, such as copyrights and patents. The one additional procedure that is added to protect intellectual property is that the compiler (e.g. a compiler 68 shown in **Figure 5**) generates encrypted byte codes. When a user attempts to run the code on the platform operating system ("OS") the verification procedures are followed as described above with reference to **Figures 6(a)** and **6(b)**. However, with the code encrypted, the operating system requires an additional approval before it is permitted to run the code. A cryptographic key is required which essentially results in an IP license to run the code. After authenticating the code, the operating system requests the IP license. The operating system verifies that the IP license is signed by the person who authored the code, and then proceeds to decrypt and execute the code. A further feature of the present invention is that third parties do not have the ability to inspect the code since it is encrypted.

Accordingly, the present invention has disclosed a method and apparatus for enhancing software security. Although the present invention has been described with reference to **Figures 1-6**, it will be apparent that may alternatives, modifications and variations may be made in light of the foregoing description.

Claims

1. A method for enhancing software security, comprising the steps of:

providing a first computer;

providing a private key;

providing an application writer's license which contains a public key;

providing software;

providing said first private key, said application writer's license and said software to a compiler executed by said first computer, said compiler compiling said software into binary code and computing a message digest for said binary code;

said first computer further encrypting said message digest using said private key, said encrypted message digest comprising an application writer's digital signature;

said first computer generating a software passport comprising said application writer's digital signature and said application writer's license;

providing an element for performing the step of distributing said software passport and said binary code to a user.

2. The method as defined by Claim 1 further including the step by said user of receiving said software passport and executing said binary code on a second computer in conjunction with an operating system.

3. The method as defined by Claim 2 wherein said license includes a name of an author of said software.

4. The method as defined by Claim 3 wherein said license further includes a public key for said author.

5. The method as defined by Claim 4 wherein said license includes a validity date for said software.

6. The method as defined by Claim 3 wherein said license further includes a digital signature for a Software Publishing Authority ("SPA").

7. A method for enhancing software security, comprising the steps of:

providing a first computer;

providing an application writer's private key;

providing an application writer's license including an application writer's public key and a platform builder's digital signature;

providing software;

providing said application writer's private key, said application writer's license and said software to a compiler executed by said first computer, said compiler compiling said software into binary code and computing a first

message digest for said binary code;

said first computer further encrypting said first message digest using said application writer's private key, said encrypted first message digest comprising an application writer's digital signature;

said first computer generating a software passport comprising said application writer's digital signature, said application writer's license and said binary code;

providing an element for performing the step of distributing said software passport to a second computer.

8. The method as defined by Claim 7 further including the steps of:

said second computer receiving said software passport;

said second computer determining if said software passport includes said application writer's license, such that if said software passport does not include said application writer's license said second computer rejects said software passport.

9. The method as defined by Claim 8 further including the step of extracting said application writer's license from said software passport.

10. The method as defined by Claim 9 further including the step of determining if said application writer's license includes said platform builder's digital signature, such that if said platform builder's digital signature is not included said software passport is rejected by said second computer.

11. The method as defined by Claim 10 further including the step of decrypting said platform builder's digital signature using a platform builder's public key provided to said second computer.

12. The method as defined by Claim 11 further including the step of said second computer computing a second message digest of said software passport and comparing said first message digest to said second message digest, such that if said first and second message digest are not equal said software passport is rejected by said second computer.

13. The method as defined by Claim 12 further including the step that if said first and second message digests are equal said second computer extracts said application writer's public key from said application writer's license.

14. The method as defined by Claim 13 further including the step of said second computer extracting said binary code from said software passport.

15. The method as defined by Claim 14 further including the step of said second computer extracting said application writer's digital signature from said software passport.

16. The method as defined by Claim 15 further including the step of said second computer computing a message digest of said binary code.

17. The method as defined by Claim 16 further including the step of said second computer decrypting said application writer's digital signature using said application writer's public key.

18. The method as defined by Claim 17 further including the step of said second computer comparing said message digest of said binary code with said decrypted application writer's digital signature, such that if said message digest of said binary code and said decrypted application writer's signature are equal, said second computer executes said binary code.

19. A system for enhancing software security, comprising:

a first computer;

a private key;

an application writer's license;

software;

a compiler executed by said first computer, said compiler compiling said software into binary code and computing a message digest for said binary code;

said first computer further encrypting said message digest using said private key, said encrypted message digest comprising an application writer's digital signature;

said first computer generating a software passport comprising said application writer's digital signature and said application writer's license;

an element for distributing said software passport and said binary code to a user.

20. The system as defined by Claim 19 wherein said user receives said software passport and executes said binary code on a second computer in conjunction with an operating system.

21. The system as defined by Claim 20 wherein said license includes a name of an author of

said software.

22. The system as defined by Claim 21 wherein said license further includes a public key for said author.

23. A system for enhancing software security, comprising:

a first computer;

an application writer's private key;

an application writer's license including an application writer's public key and a platform builder's digital signature;

software;

a compiler executed by said first computer, said compiler compiling said software into binary code and computing a first message digest for said binary code;

said compiler further encrypting said first message digest using said application writer's private key, said encrypted first message digest comprising an application writer's digital signature;

said first computer generating a software passport comprising said application writer's digital signature, said application writer's license and said binary code;

an element for distributing said software passport to a second computer.

24. The system as defined by Claim 23 wherein said second computer receives said software passport and determines if said software passport includes said application writer's license, such that if said software passport does not include said application writer's license said second computer rejects said software passport.

25. The system as defined by Claim 24 wherein said second computer extracts said application writer's license from said software passport.

26. The system as defined by Claim 25 wherein said second computer determines if said application writer's license includes said platform builder's digital signature, such that if said platform builder's digital signature is not included said software passport is rejected by said second computer.

27. The system as defined by Claim 26 wherein said second computer decrypts said platform builder's digital signature using a platform builder's public key provided to said second computer.

28. The system as defined by Claim 27 wherein said second computer computes a second message digest of said software passport and compares said first message digest to said second message digest, such that if said first and second message digest are not equal said software passport is rejected by said second computer.

29. The system as defined by Claim 28 wherein if said first and second message digests are equal said second computer extracts said application writer's public key from said application writer's license.

30. The system as defined by Claim 29 wherein said second computer extracts said binary code from said software passport.

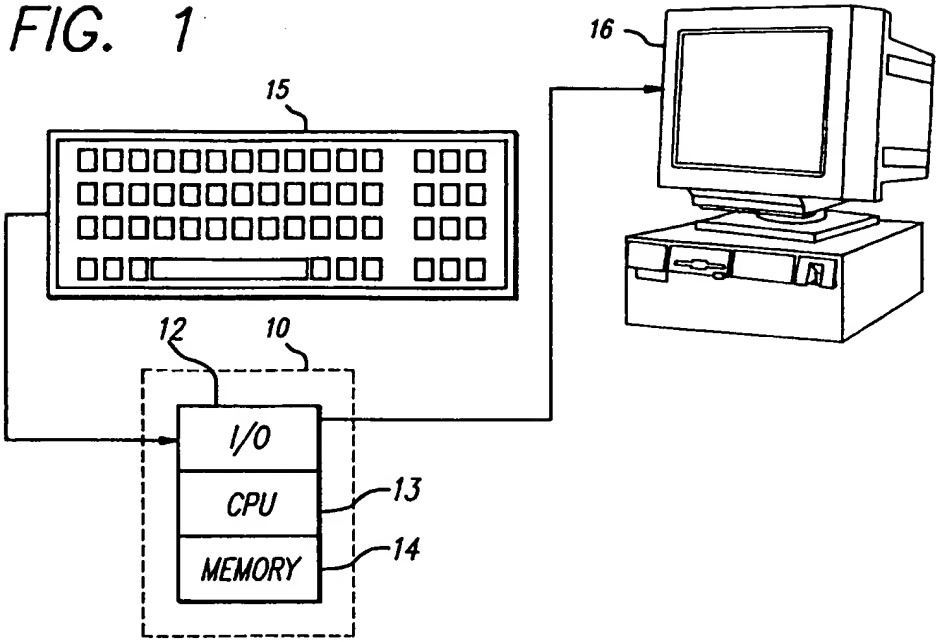
31. The system as defined by Claim 30 wherein said second computer extracts said application writer's digital signature from said software passport.

32. The system as defined by Claim 31 wherein said second computer computes a message digest of said binary code.

33. The system as defined by Claim 32 wherein said second computer decrypts said application writer's digital signature using said application writer's public key.

34. The system as defined by Claim 33 wherein said second computer compares said message digest of said binary code with said decrypted application writer's digital signature, such that if said message digest of said binary code and said decrypted application writer's signature are equal, said second computer executes said binary code.

FIG. 1


16
15
12
10
I/O
CPU
MEMORY
13
14

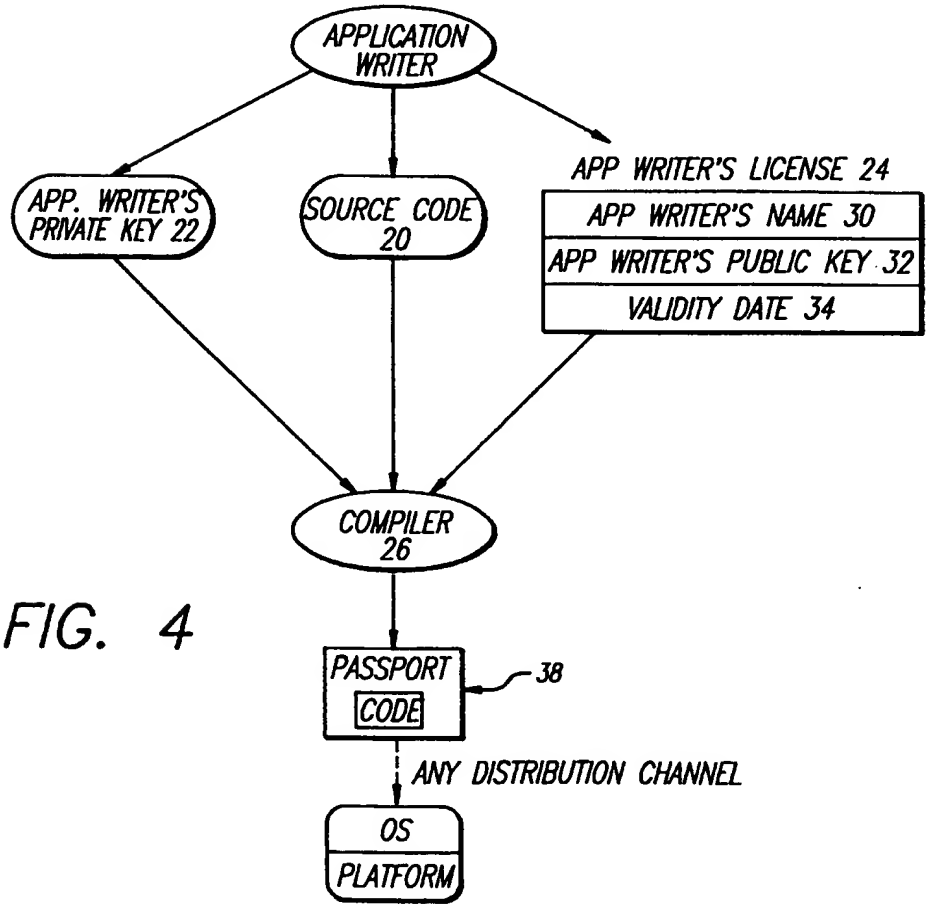
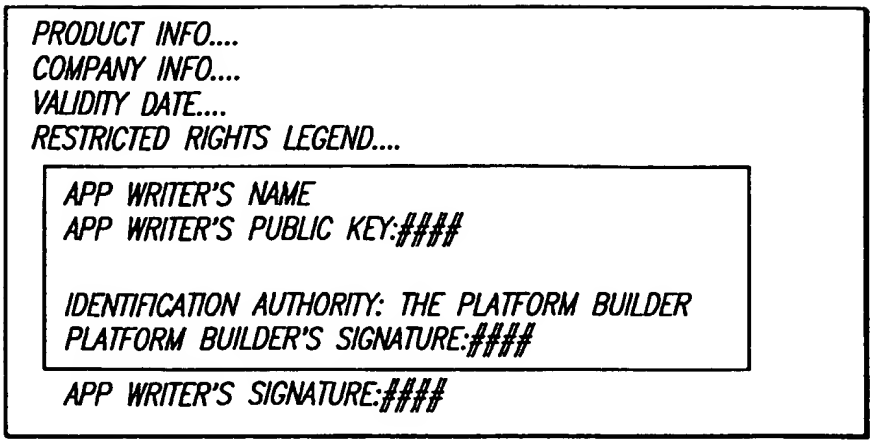

APPLICATION WRITER
APP. WRITER'S PRIVATE KEY 22
SOURCE CODE 20
APP WRITER'S LICENSE 24
APP WRITER'S NAME 30
APP WRITER'S PUBLIC KEY 32
VALIDITY DATE 34
COMPILER 26
PASSPORT
CODE
38
ANY DISTRIBUTION CHANNEL
OS
PLATFORM

FIG. 4

FIG. 2


PRODUCT INFO....
COMPANY INFO....
VALIDITY DATE....
RESTRICTED RIGHTS LEGEND....
APP WRITER'S NAME
APP WRITER'S PUBLIC KEY:####
IDENTIFICATION AUTHORITY: THE PLATFORM BUILDER
PLATFORM BUILDER'S SIGNATURE:####
APP WRITER'S SIGNATURE:####
CODE BODY

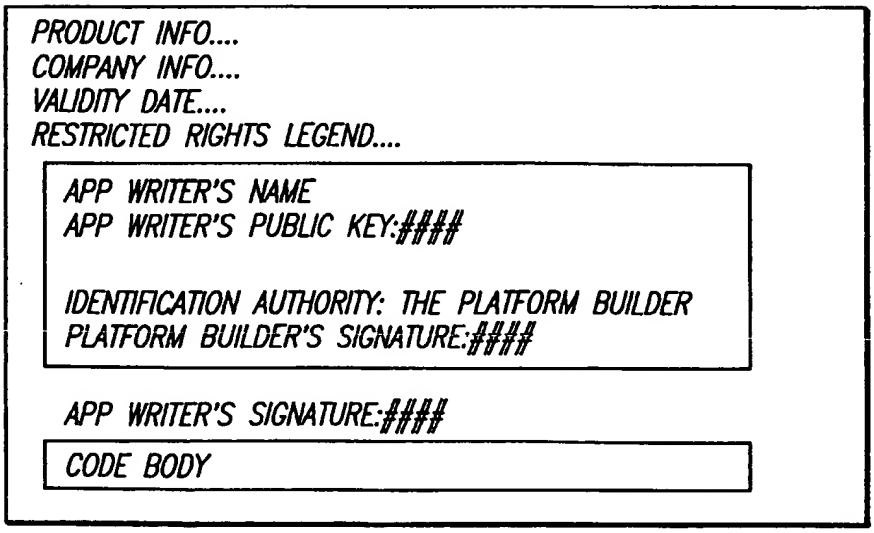

PRODUCT INFO....
COMPANY INFO....
VALIDITY DATE....
RESTRICTED RIGHTS LEGEND....
APP WRITER'S NAME
APP WRITER'S PUBLIC KEY:####
IDENTIFICATION AUTHORITY: THE PLATFORM BUILDER
PLATFORM BUILDER'S SIGNATURE:####
APP WRITER'S SIGNATURE:####
CODE BODY

FIG. 3

FIG. 5

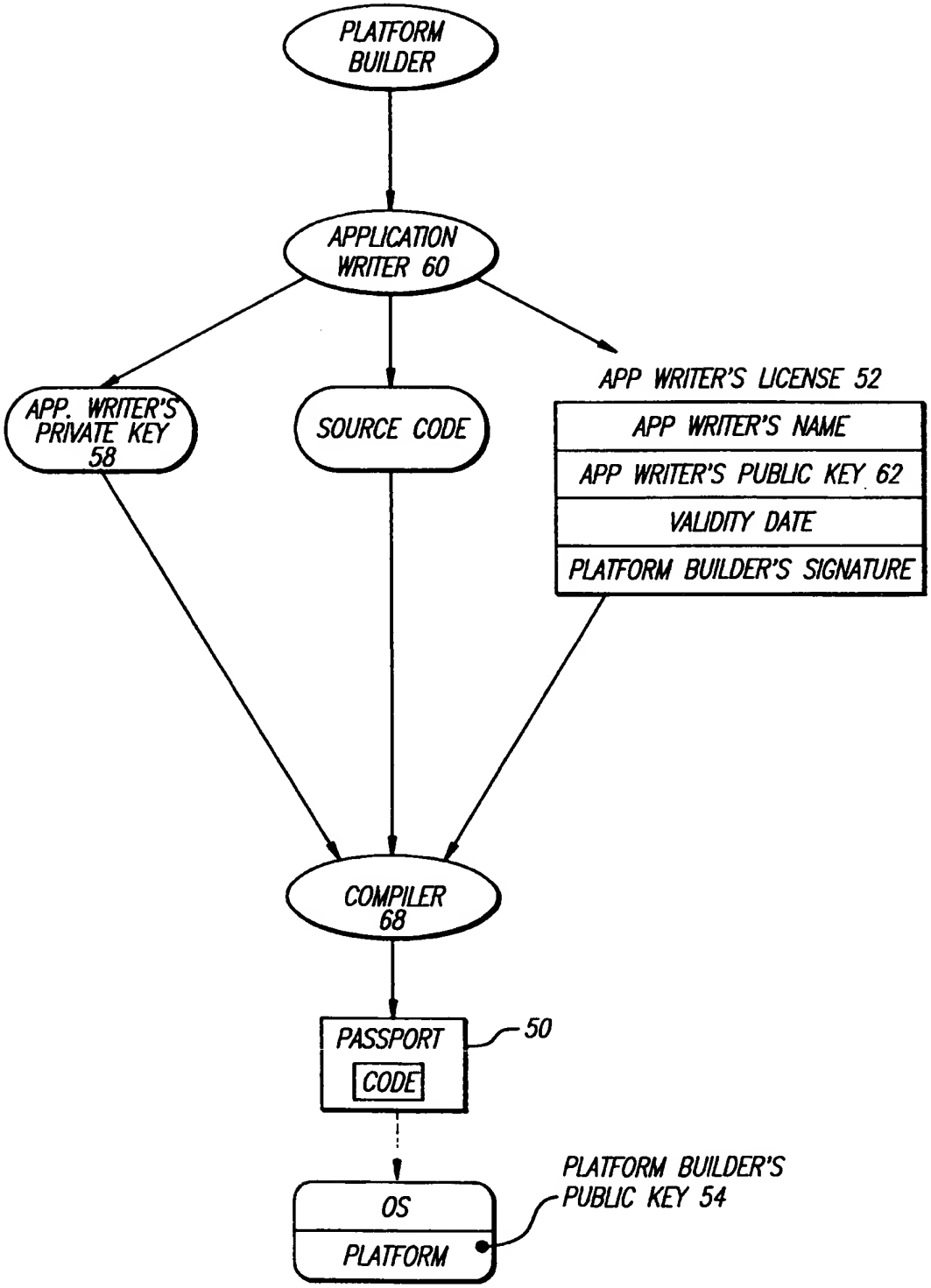
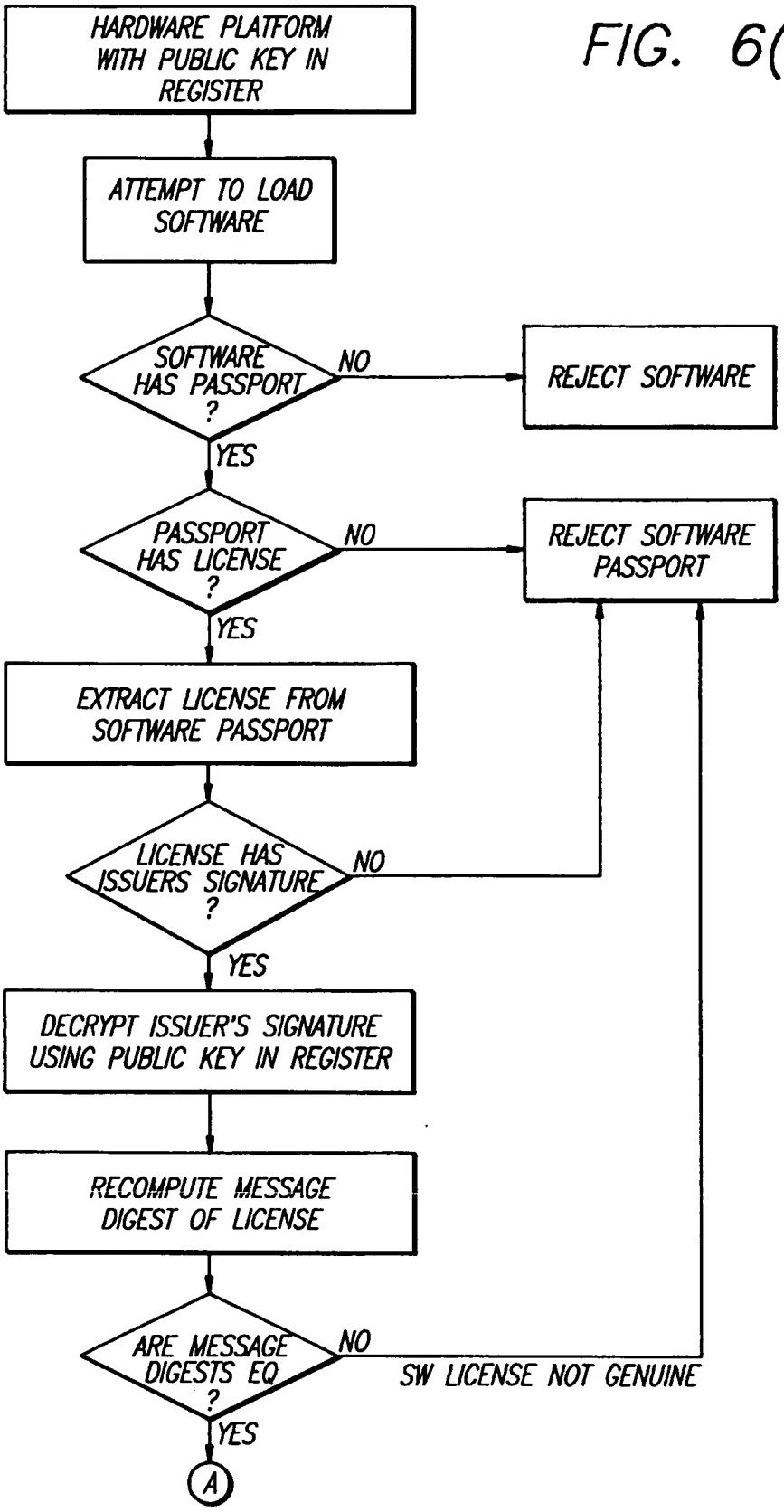

PLATFORM BUILDER
APPLICATION WRITER 60
APP. WRITER'S PRIVATE KEY 58
SOURCE CODE
APP WRITER'S LICENSE 52
APP WRITER'S NAME
APP WRITER'S PUBLIC KEY 62
VALIDITY DATE
PLATFORM BUILDER'S SIGNATURE
COMPILER 68
PASSPORT
CODE
50
OS
PLATFORM
PLATFORM BUILDER'S PUBLIC KEY 54

FIG. 6(a)


HARDWARE PLATFORM WITH PUBLIC KEY IN REGISTER
ATTEMPT TO LOAD SOFTWARE
SOFTWARE HAS PASSPORT ?
NO
REJECT SOFTWARE
YES
PASSPORT HAS LICENSE ?
NO
REJECT SOFTWARE PASSPORT
YES
EXTRACT LICENSE FROM SOFTWARE PASSPORT
LICENSE HAS ISSUERS SIGNATURE ?
NO
YES
DECRYPT ISSUER'S SIGNATURE USING PUBLIC KEY IN REGISTER
RECOMPUTE MESSAGE DIGEST OF LICENSE
ARE MESSAGE DIGESTS EQ ?
NO
SW LICENSE NOT GENUINE
YES
A

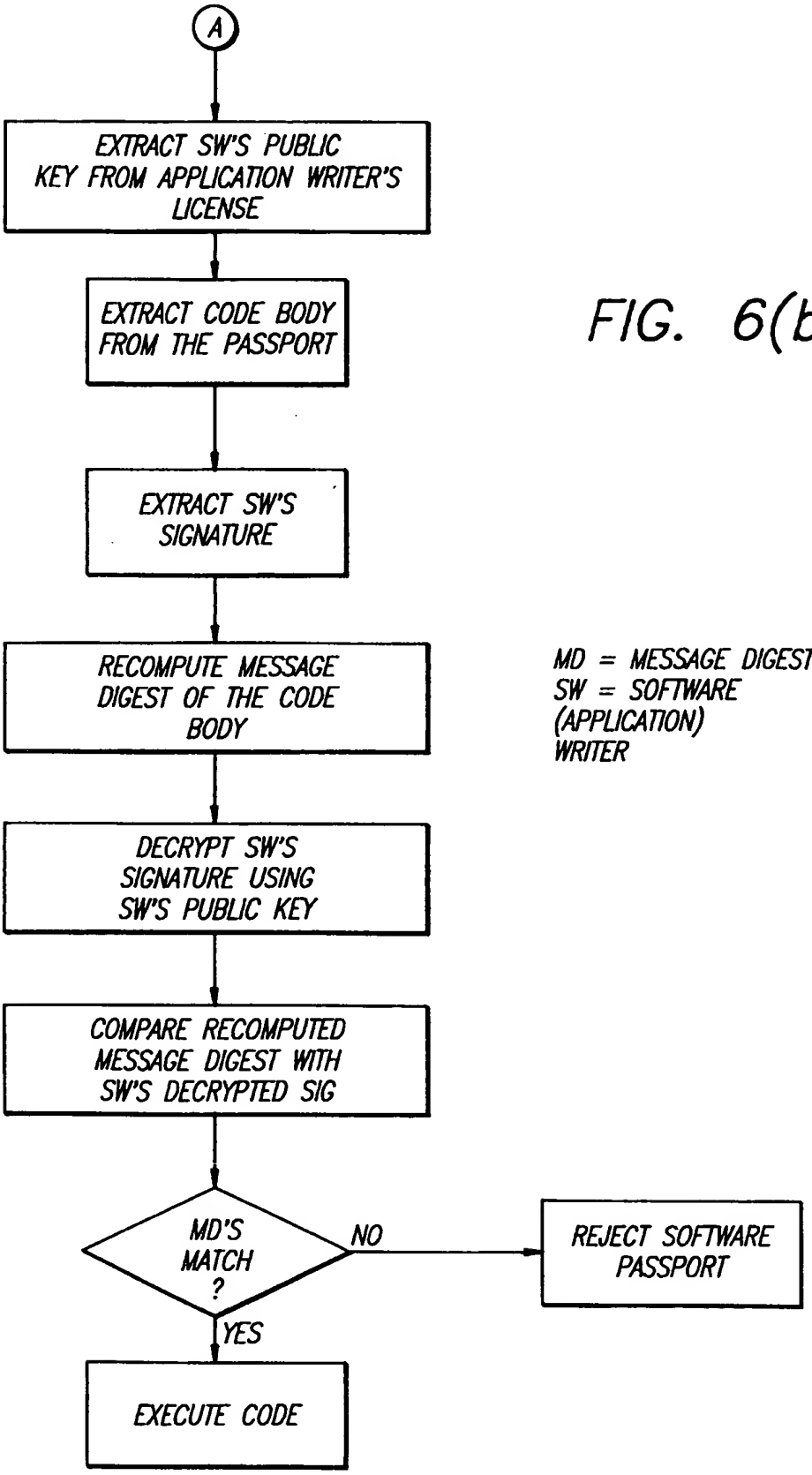


FIG. 6(b)

MD = MESSAGE DIGEST
SW = SOFTWARE
(APPLICATION)
WRITER